RL-TR-95-198
Final Technical Report
October 1995

AN ADVANCED ARCHITECTURE FOR INTELLIGENT FINITE-ELEMENT MODELING AND ANALYSIS

University of Massachusetts

D. Corkill and V. Manakkal

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

APPROVED: Dale W. Richards

DALE W. RICHARDS
Project Engineer

FOR THE COMMANDER: John J. Bart

JOHN J. BART
Chief Scientist, Reliability Sciences
Electromagnetics & Reliability Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE October 1995	3. REPORT TYPE AND DATES COVERED Final Jan 94 - Jan 95

4. TITLE AND SUBTITLE

AN ENHANCED ARCHITECTURE FOR INTELLIGENT FINITE-ELEMENT MODELING AND ANALYSIS

5. FUNDING NUMBERS

C - F30602-93-C-0085
PE - 62702F
PR - 2338
TA - 02
WU - PG

6. AUTHOR(S)

D. Corkill and V. Manakkal

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Massachusetts
Department of Computer Science
Amherst MA 01002

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory (ERSR)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-95-198

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Dale W. Richards/ERSR/(315) 330-3476

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes research to improve the capabilities and performance of the Rome Laboratory Intelligent Module Analyzer (IMCMA). (See "An Architecture for Intelligent Multichip Module Reliability Analysis," RL-TR-94-71.) IMCAM is a blackboard-based software tool that automatically applies finite-element and knowledge-based analysis to rapidly assess the thermal reliability of microelectronic multichip module (MCM) designs. IMCMA is a cooperative effort that involves Rome Laboratory, the Mechanical Engineering and Computer Science Departments at the University of Massachusetts, and Blackboard Technology Group, Inc. In IMCMA, the amount of modeling effort and expertise required of the designer has been reduced through: (1) the use of a high-level representation of devices as the interface between the designer and the analysis tools; and (2) incorporating the modeling and analysis expertise of experienced design analysts into the software, making it available to less experienced designers. In this effort, the previous architecture was extended to represent finite-element mesh objects and results directly on the blackboard, and to provide separate MCM defined and "as modeled" descriptions. In addition, graphical facilities for editing and displaying MCM descriptions and material specifications were developed along with an improved interactive display of the analysis results.

14. SUBJECT TERMS

Multichip module, MCM, Blackboard, Artificial intelligence, Reliability, Thermal analysis, Computer-aided design, Finite element analysis, FEA, Modeling

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Executive Summary

This report describes the results of our one-year effort in enhancing a prototype architecture for an intelligent, automated analysis system that can be used to provide rapid reliability assessments of multichip microelectronic module designs. This effort is part of a cooperative effort between the Design Analysis Branch at Rome Laboratory and the Mechanical Engineering and Computer Science Departments at the University of Massachusetts in developing a powerful computer-based modeling and analysis system called the *Intelligent Multichip Module Analyzer* (IMCMA). The IMCMA system is a blackboard-based software tool that automatically applies finite-element and knowledge-based analysis to rapidly assess the reliability of microelectronic multichip module (MCM) designs. The software bases its evaluation on environmentally and operationally induced failure mechanisms. It is useful in assessing the reliability of advanced microelectronic devices that have no historical reliability data.

Extensive understanding of the analysis and the failure prediction of advanced electronics is being incorporated into IMCMA. Rome Laboratory is the Air Force's center of expertise for the reliability assessment of advanced electronics and is experienced in the design, construction, and analysis of MCMs, especially in the areas of MCM failure mechanisms and reliability assessment. The Mechanical Engineering Department at UMass provides significant expertise in finite-element modeling and analysis and the Computer Science Department provides expertise in knowledge-based problem solving, problem-solving representations, and control.

Early detection of design-related reliability problems can significantly decrease the development, manufacture, and testing costs of MCMs. This potential savings is offset by the labor-intensive nature of modeling as a means of detecting design-related reliability problems. Modeling an MCM requires significant expertise to build and, if necessary, remodel critical regions of an initial finite-element model. Typically, it may take a senior design analyst several weeks to construct and analyze an initial computer-based model of a microelectronic device. An expert design analyst must decide how to model the individual components, how to represent them so they can be used by analysis tools (such as an automated mesh generator), and how to interpret the results. The numerical results from the initial model will often indicate critical regions of the device that must be remodeled in order to obtain an accurate analysis of the mechanical stresses of these regions.

Current analysis tools lack high-level models of microelectronic devices and loading conditions. Based on low-level geometric representations,

these tools require a skilled expert to translate the high-level model of the device into representations that the analysis tools can utilize. Conversion between different representations used in each tool is also often left to the designer.

In the IMCMA system, modeling effort and expert-operator requirements have been reduced through two main techniques:

- Use of a high-level representation of devices as the interface between the designer and the analysis tools
- Capturing the expertise of experienced design analysts in an intelligent assistant to be made available to less experienced designers

Increasing the productivity of design experts significantly reduces the development effort, lead time, and cost associated with new MCMs.

In this effort, we extended the original IMCMA prototype to represent finite-element mesh objects and results directly on the blackboard, provided separate device description and device "as modeled" representations, developed graphical facilities for creating, modifying, copying, and deleting MCM-device components and material specifications, and redesigned and implemented an improved interactive result-display facility. In this report, we describe our findings and progress.

Acknowledgments

The IMCMA project is a cooperative effort involving a number of individuals from Rome Laboratory and the Mechanical Engineering and Computer Science Departments at the University of Massachusetts at Amherst. Without the friendly and open interchange of ideas and expertise among everyone involved with the IMCMA effort, the research described in this report would not have been accomplished.

From the UMass Mechanical Engineering Department, Professor Ian Grosse and his students maintained the Sandia finite-element-generation tools and the FEECAP finite-element analysis code used in IMCMA.

Table of Contents

1 Introduction

Early detection of design-related reliability problems can significantly decrease the development, manufacture, and testing costs of multichip microelectronic modules (MCMs). For example, the Design Analysis Branch (ERSD) of the Electromagnetic Reliability Directorate at Rome Laboratory has successfully used the finite-element method for reliability assessment of MCM components to predict failure modes and assess their mechanical reliability [1,2]. Detailed simulation studies of microelectronic devices has been used to successfully predict the location and magnitude of critical thermal and physical stresses [3,4,5,6,7,8]. These studies can be used to predict the reliability and failure modes of the device.

Detecting design-related reliability problems using detailed simulation studies is labor intensive, and requires significant expertise to build and, if necessary, remodel critical regions of an initial finite-element model of MCM devices on the computer. Typically, it may take an engineer several weeks to construct and analyze an initial model of a microelectronic device on the computer. The numerical results from the initial model will often indicate critical regions of the device which must be remodeled in order to obtain an accurate model of these regions. For example, remeshing is needed to resolve meshing problems due to violation of geometric transitioning constraints or due to basic limitations of the mesh generator. Increasing the productivity of design experts will significantly reduce the development effort, lead time, and cost associated with new MCMs.

The goal of IMCMA is to reduce modeling effort and expert operator requirements through several techniques:

- Use of a high-level representation of devices as the interface between the designer and the analysis tools. The designer would define the device in terms of its components and the analysis system would use these high-level definitions to develop a detailed representation of the device. For example, a MCM might consist of a number of uniform rectangular chips and capacitors mounted on the substrate in a symmetrical pattern. The designer would specify the chips, capacitors, and the pattern, and the system would develop all the detailed submodels of critical features of the device.
- By capturing the expertise of experienced designers in an intelligent assistant for MCM device analysis, much of the labor-intensive aspects of reliability analysis can be automated and made available to less experienced designers [9]. Instead of

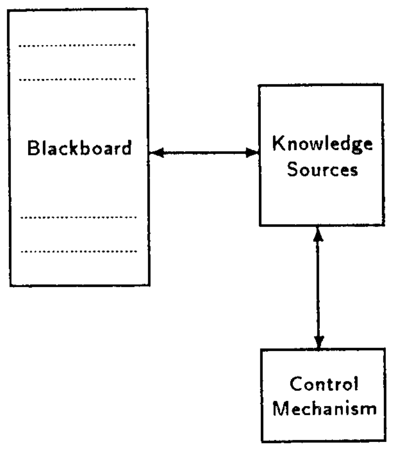


Figure 1 Basic Components of the Blackboard Model

> the expert designer deciding how to use the tools to effectively model devices, the analysis system would develop and implement a modeling strategy for analyzing the device. The system would monitor the accuracy of the modeling process, detecting modeling problems such as idealizations resulting in singularities in the finite-element solution, violations of mesh transitioning constraints, and poorly structured meshes. Once detected, the system would reformulate the model or remesh until an acceptable model is developed.

These techniques form the basis of the IMCMA system.

2 An Overview of the IMCMA Prototype

This effort augments prior IMCMA-development efforts conducted at Rome Laboratory and the University of Massachusetts. At the start of this effort, the prototype IMCMA was at a stage where a high-level device specification can be processed through model simplification, finite-element generation, and thermal analysis. In this section we briefly describe the state of the IMCMA prototype at the start of this effort.

In the prior IMCMA efforts, a blackboard system, based on the blackboard problem-solving paradigm [10,11], was selected as the basis for the IMCMA architecture. A blackboard system performs problem solving by using three basic components (Figure 1):

- A *blackboard*, that is a global database containing input data, partial solutions, and other data that are in various problem-solving states.
- *Knowledge sources* (KSs) which are independent modules that contain the knowledge needed to solve the problem, and that can be widely diverse in representation and in inference techniques. KS modularity facilitates application development and simplifies maintenance and enhancement.
- A *control mechanism*, that is separate from the individual KSs and that makes dynamic decisions about which KS is to be executed next.

The IMCMA prototype is implemented using Blackboard Technology Group, Inc.'s GBB™ framework, a toolkit for rapidly developing and delivering high-performance blackboard applications. GBB provides the infrastructure for a blackboard-based application such as IMCMA, as well as a graphical user interface toolkit and graphical monitoring and inspection tools.

Original IMCMA Knowledge Sources

The original IMCMA prototype contained nine KSs (Table 1). Processing proceeded among these KSs as shown in Figure 2.

Status of the Original IMCMA Prototype

The original IMCMA prototype effort resulted in the following:

- **Object class specifications** — A high-level, object-oriented, hierarchical representation for MCM devices and components was developed. This representation simplified specification of MCM devices and includes intermediate data objects needed by the various KSs during their computations.
- **Initial code interfaces** — The blackboard system and KSs needed to interact with existing software codes and codes being developed by other IMCMA researchers. Interfaces have been developed using formatted file transfer between the blackboard and the codes. Although less efficient than a direct application interface (API), a file-based approach allowed the existing formatted file structure present in some codes to be used directly.

GBB is a trademark of Blackboard Technology Group, Inc.

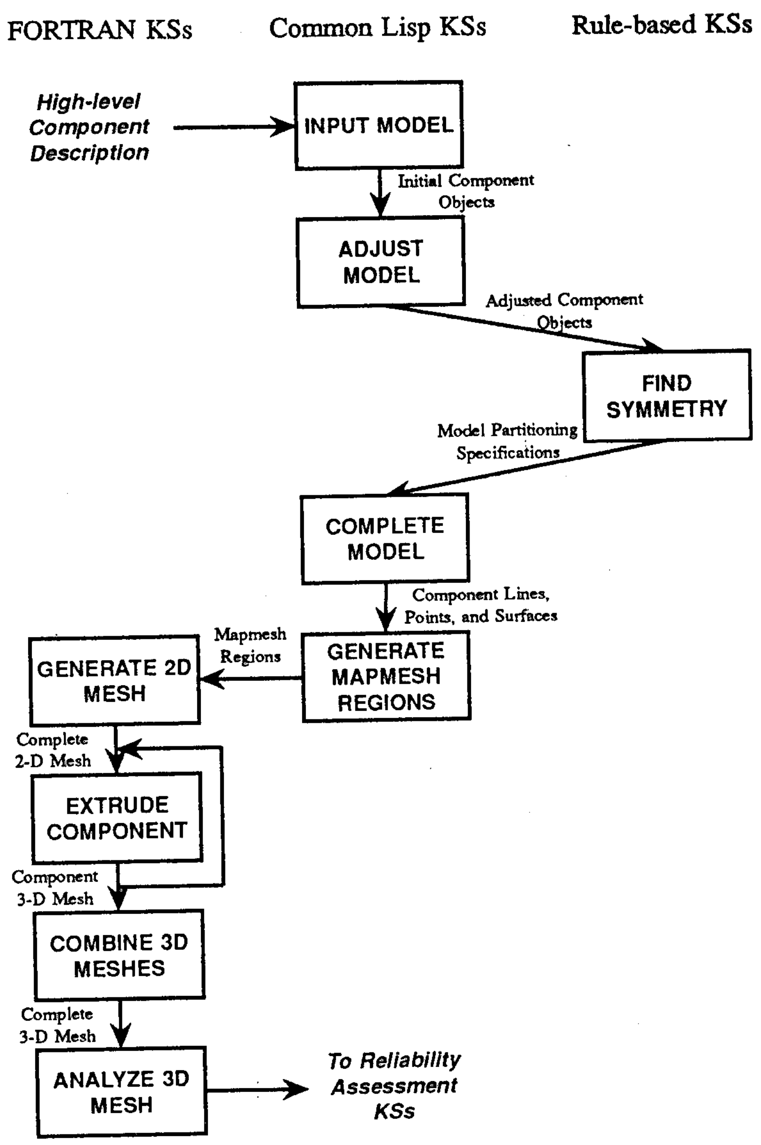


Figure 2 Original IMCMA KS Processing

`input-model`	GBB	Reads a device specification file and builds the component and material objects specified therein
`adjust-model`	GBB	Simplifies the geometry of the device to facilitate rapid analysis
`find-symmetry`	GBB/CLIPS	Identifies 2D (XY) geometric symmetries in the device
`complete-model`	GBB	Builds the component objects that are based on the adjusted model
`generate-mapmesh-regions`	GBB	Generates the coarsest possible 2D mesh for the adjusted device
`generate-2d-mesh`	GBB/FORTRAN	Generates a 2D mesh from the mapmesh regions and mesh density specifications
`extrude-component`	GBB/FORTRAN	Edits the 2D mesh for a specific component and extrudes that component into a 3D mesh
`combine-3d-meshes`	GBB/FORTRAN	Combines all the component 3D meshes into a single 3D mesh
`analyze-3d-mesh`	GBB/FORTRAN	Analyzes the combined 3D mesh

Table 1 Original IMCMA Knowledge Sources

- **Simple graphical-output facilities** — The use of GBB has provided simple graphical-output facilities, allowing for two-dimensional plotting of multidimensional modeling and analysis data.

3 Enhancements to the Original IMCMA Prototype

In this effort, we extended and enhanced the prototype IMCMA system in the following ways:

- **User interface.** Using the prototype IMCMA prototype system requires learning the device-definition file syntax (Appendix A) and the commands needed to perform an analysis. In practice, a system such as IMCMA will not be used unless it is easy to use by designers and analysts. The most important user-interface areas are the specification and modification of high-level device descriptions and presentation of the results of analysis. These two areas were directly addressed in this effort.
- **Data sharing.** For analysis and submodeling purposes, it is important that the results of finite-element modeling be made available to all KSs. This means that the results of the various KS

codes cannot be retained within the individual tools or external Exodus files, but must be placed on the blackboard so that all KSs can have access to them. In this effort we added the ability to read the Exodus binary-file meshing data produced by the `FASTQ` and `GEN3D` KSs, to place individual 2D and 3D element and node objects onto the blackboard. This now allows KSs to use the advanced retrieval capabilities of the underlying GBB architecture in performing categorization and analysis activities.

In the following sections, we describe in further detail these IMCMA enhancements.

Representational Restructuring

In this effort, several important changes were made to the blackboard and object representations used in the original IMCMA prototype. The biggest change involved separating the component-level device representation from that used to model and analyze the device. This separation provides flexibility in defining and modifying the device definition prior to (or following) analysis.

Separation of device definition from device model

In the original IMCMA prototype, blackboard objects were created from the device-description file to represent the MCM device's components and materials. These `component` and `material` objects were then used to generate the 3D mesh and were subject to modeling simplifications made during the analysis.

Although this representation worked well in the prototype architecture, it presents problems in creating a clean distinction between the device as defined and the device as modeled. For example, because the complete device definition was used for developing the model, performing a partial (spatial) analysis of the device would require the specification of an artificial partial device. Similarly, performing an analysis of a device with a modified material property (for example, changing the thermal conductivity property of a material to represent a uniform voiding) required changing the actual property of the defined material.

A second problem with the single representation in the IMCMA prototype was the difficulty of editing the original device definition for reanalysis (either manually or under automatic control of programs such as the design-of-experiments (DOVE) shell). Since the IMCMA prototype was free to change attributes of the original device's components (such as shifting components slightly to simplify analysis) the original device information could be lost during modeling and could only be recreated by re-reading the device definition information.

In this effort, we separated the original representation into two:

- `defined-components` and `defined-materials`—represent the actual device definition
- `modeled-components` and `modeled-materials`—represent the device as modeled

By first creating the `defined-components` and `defined-materials`, either from the device-definition file or by using the graphical user interface (discussed later in this report), the enhanced IMCMA system maintains a "protected" description of the device under analysis.

A new KS (`create-model-ks`) was written to create a modeling representation of the device from the `defined-components` and `defined-materials` (Table 2). If desired, this KS could be written to copy only a spatial portion of the device, to exclude specific types of components (such as glue layers), or to modify material properties. The IMCMA system remained free to modify the `modeled-components` and `modeled-materials` as needed, and a new modeling and analysis could still be performed using the original `defined-components` and `defined-materials`. Processing proceeds among KSs in the enhanced IMCMA prototype as shown in Figure 3.

Power-dissipation surfaces

Missing in the original IMCMA prototype is the ability to represent power-dissipation surfaces, such as those used to model the active electrical components on a chip. Power dissipation surfaces are idealized heat-producing surfaces used to model the thermally active portions of active MCM components. The FEECAP mesh-analysis program is able to analyze surfaces with a prespecified flux (directional flow), but not surfaces with an unspecified heat flow. In other words, if power-dissipation surfaces were represented as surfaces with heat flux, the analysis would be incorrectly account for a power-dissipation surface if there are hotter regions nearby.

In this effort, we converted each power dissipation surface (assigned to a specific face of a component) to a set of point heat sources. The value assigned to each point heat source was apportioned based on the surface area of the power dissipation surface in the neighborhood of the point heat source. This approach is a modeling approximation, but one that is sufficiently accurate for analysis purposes.

`input-model`	GBB	Reads a device specification file and builds the defined-component and defined-material objects specified therein
`create-model`	GBB	Creates modeled component and material objects from the defined-component and defined-material objects present on the blackboard
`adjust-model`	GBB	Simplifies the geometry of the device to facilitate rapid analysis
`find-symmetry`	GBB/CLIPS	Identifies 2D (XY) geometric symmetries in the device
`complete-model`	GBB	Builds the component objects that are based on the adjusted model
`generate-mapmesh-regions`	GBB	Generates the coarsest possible 2D mesh for the adjusted device
`generate-2d-mesh`	GBB/FORTRAN	Generates a 2D mesh from the mapmesh regions and mesh density specifications
`extrude-component`	GBB/FORTRAN	Edits the 2D mesh for a specific component and extrudes that component into a 3D mesh
`combine-3d-meshes`	GBB/FORTRAN	Combines all the component 3D meshes into a single 3D mesh
`analyze-3d-mesh`	GBB/FORTRAN	Analyzes the combined 3D mesh

Table 2 Updated IMCMA Knowledge Sources

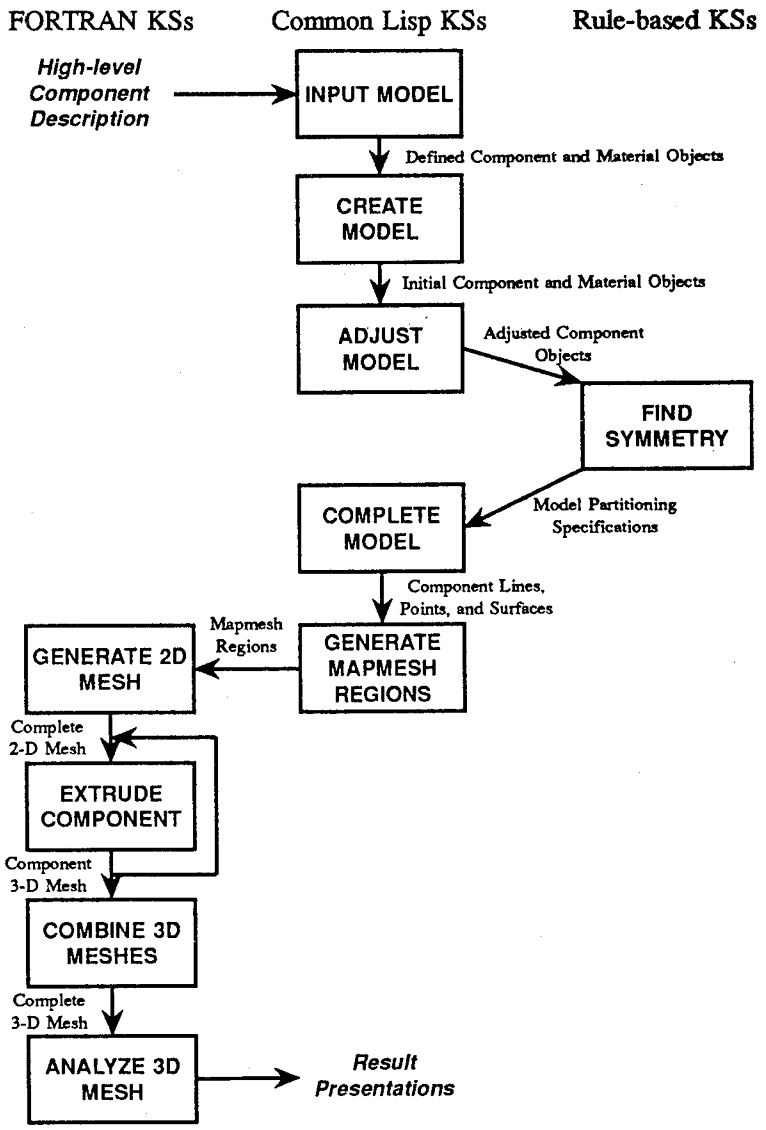


Figure 3 Enhanced IMCMA KS Processing

Robust Exodus-file reader/writer

The Sandia mesh-generation tools used in the original IMCMA prototype exchange data using binary files known as Exodus files. In order to achieve the goal of representing the individual 2D and 3D element and node objects on the blackboard, it was necessary to write an Exodus file reader in Common Lisp, so that the information generated by these FORTRAN-based tools could be placed on the blackboard for further analysis.

Reading FORTRAN binary data files

Common Lisp does not provide a direct facility for reading the FORTRAN binary files used in the Exodus file representation. This meant that the basic routines to read the Exodus binary files had to be developed.

Functions for reading both integer and floating-point numerical data from the binary files were written. We began by attempting to read integer data from the Exodus files. Common Lisp provides a primitive `read-byte` function that reads a binary-byte quantity from a file stream. Our first approach was to use the Common Lisp function `ldb` to replace the bytes within a Common Lisp integer with the bytes that were read from the binary file. This approach worked for integer data, but would not work for floating point data, as `ldb` only works for integer targets and Common Lisp (properly) does not allow coercion of a fixnum to a floating-point value.

One might attempt to displace a single-element integer array to a single-element floating point array (assuming single precision floating point values, such as in Exodus). However, this approach violates Common Lisp standards for displayed arrays.

Essentially, the Common Lisp standard does not provide a direct means for obtaining binary floating-point values from an input stream[1]. To implement the Exodus input reader, we escaped the Common Lisp type-mangling constraints by using Allegro Common Lisp's foreign-data-structure facilities. We defined C-compliant data structures for both integer and single- and double-float values (Figure 4). A static foreign data structure was allocated for each of the three data types, allowing the individual 8-bit `read-byte` values to be stored into the appropriate data structure using the unsigned-type accessors and to be retrieved as the appropriate data type using the appropriate integer or floating point accessor. These `read-integer`, `read-single-float`, and `read-double-float` operators formed the basic building blocks of the Exodus file reader.

[1]Actually, the standard does include the function `read-sequence`, but at the time of this effort, Common Lisp vendors had not yet implemented such a recent addition.

```
;;; C-Struct for holding single floats
(ff::def-c-type (df :in-foreign-space) :single-float)

;;; C-Struct for holding the 4 bytes in the single-float
(ff::def-c-type (ba :in-foreign-space) 4 :unsigned-byte)

;;;  C-Struct for holding double floats
(ff::def-c-type (dd :in-foreign-space) :double-float)

;;;  C-Struct for holding the 8 bytes of double floats
(ff::def-c-type (bd :in-foreign-space) 8 :unsigned-byte)

;;;  C-Struct for holding integers
(ff::def-c-type (di :in-foreign-space) :int)

;;;  C-Struct for holding 4 bytes of the integers
(ff::def-c-type (bi :in-foreign-space) 4 :unsigned-byte)

;;; ****************************************
;;; *i* integer accessor variable
;;; *x* single-float accessor variable
;;; *d* double-float accessor variable
;;; ****************************************

(defvar *i* (ff::make-cstruct 'di))
(defvar *x* (ff::make-cstruct 'df))
(defvar *d* (ff::make-cstruct 'dd))
```

Figure 4 C Structures for the Exodus File Reader

Reading the Exodus file

The Exodus file specification provided by Sandia National Laboratories provides detailed information on the order in which data is stored in the Exodus file. Despite these detailed specifications, difficulties were encountered while trying to read the file, as some of the values were found to be unreasonable. These values were caused by extra byte sequences in the file. These additional bytes are part of FORTRAN's binary file representations, used to help identify and read in data from binary files. To identify these extra bytes, a small `dump-file` function was written which could be invoked at the point in the file where additional bytes were suspected. Using `dump-file` and the FORTRAN source files that write Exodus files, the algorithm that was used to generate the contents of the additional bytes was identified.

These information bytes are byte counts written in integer format by FORTRAN at the beginning and end of each implicit `DO`-loop enclosing a `write` operation. The value of these bytes are the total number of bytes to be written for that loop. If the `DO`-loop will not write any bytes, an integer value of zero is written. Finally, if the data is of type `CHARACTER*8` and several `DO`-loops are used consecutively, the byte count is written twice, at the beginning and at the end of character data, with the value being cumulative.

The Exodus file writer

In the original IMCMA prototype, the mesh-analysis results computed by FEECAP as part of the **analyze-3d-mesh** KS are appended directly to the Exodus input file by the FEECAP program. Once the Exodus file reader was completed, the FEECAP analysis results were now available on the blackboard. This opened the possibility of adding additional IMCMA KSs to do other kinds of analysis, adding to the results present on the blackboard.

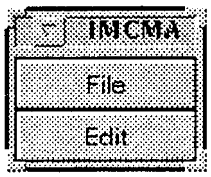
To support creating Exodus files containing results present on the blackboard, an Exodus files writer was needed. In the case of the Exodus file writer, the inverse problem of producing the appropriate byte counts for FORTRAN existed, and required knowing how the FORTRAN `DO`-loops were being used to read the binary data. Conversion from of the internal Lisp values to integer and single- and double-precision floating data was performed using the same foreign-structure approach used in the Exodus file reader.

Menu-based User Interface

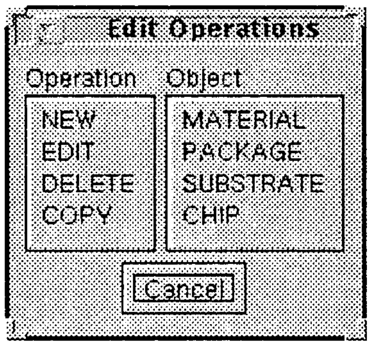
In order to use the original IMCMA prototype, a design engineer would first create a text-based *device-specification file*, containing the geometric layout of the device, the properties of the materials used in the device, and the static and thermal conditions (an example device-specification file is shown in Appendix A). Although this device-specification file provides a convenient external representation for the device, its construction requires the engineer to understand the file's syntax and to correctly type the various forms in the file.

In this effort, we augmented the IMCMA prototype with a graphics-based facility for creating and maintaining device-specification files. This facility provides four data-entry dialogs for creating, modifying, and copying materials, packages, substrates, and chips (Figures 5–8).[2]

The editing operations are selected using the **Edit** button in the IMCMA main menu:

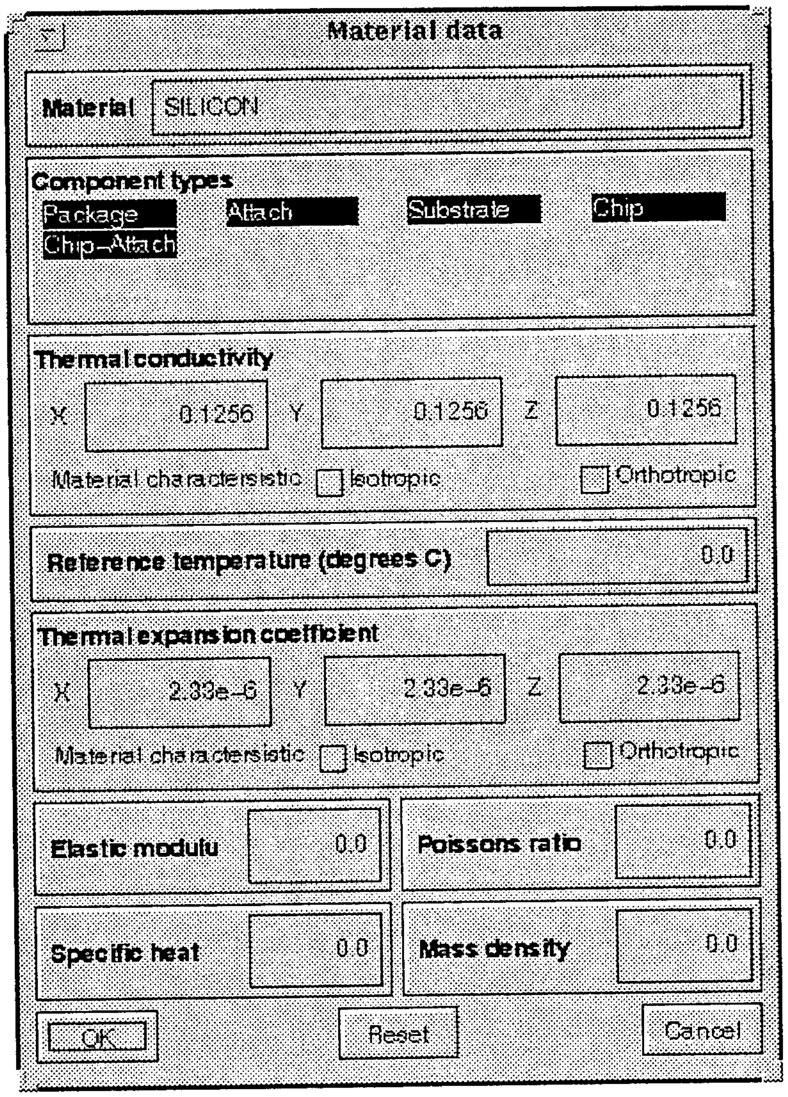


The editing operations allow creation, modification, copying, and deletion of materials and package, substrate, and chip component specifications:



The use of ChalkBox™ in conjunction with GBB's **copy-unit** and **reinitialize-instance** functions made implementing these editing operations relatively easy. When editing or copying an existing material or component, initial values are extracted from the existing component and are used to fill in the text-entry and similar widgets in the edit dialog. Each of the widgets is named the same as the initialization argument keyword to be used in **reinitialize-instance**. When the edit dialog is exited, the widget name (initialization keyword) and value are

[2]Subtrate and chip attach components are specified as part of the substrate and chip components in the graphical facility, but they are represented separately in the device-specification file. The graphical facility implemented in this effort manages the conversion between the two representations.
GBB is a trademark of Blackboard Technology Group, Inc.



Material data

Material: SILICON

Component types: Package, Attach, Substrate, Chip, Chip-Attach

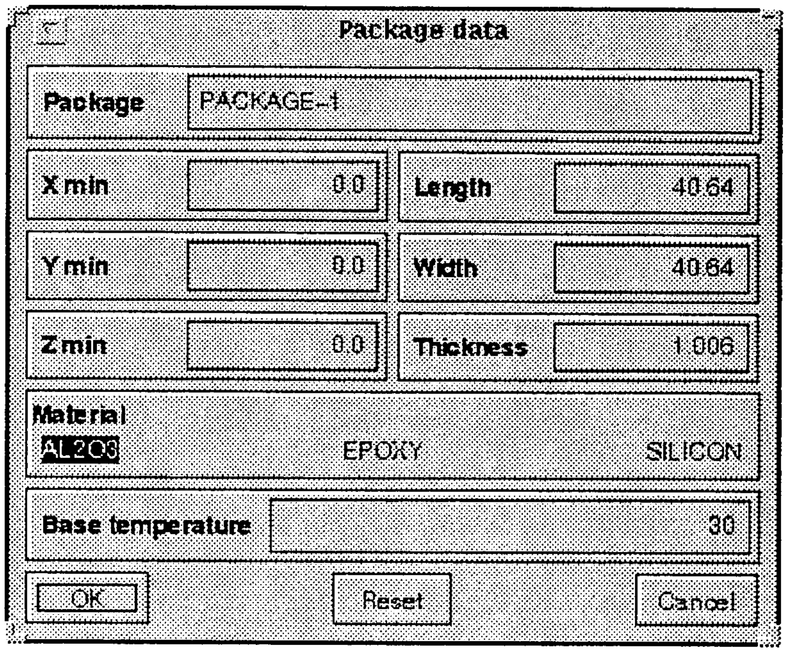
Thermal conductivity
X 0.1256 Y 0.1256 Z 0.1256
Material characteristic [ ] Isotropic [ ] Orthotropic

Reference temperature (degrees C) 0.0

Thermal expansion coefficient
X 2.33e-6 Y 2.33e-6 Z 2.33e-6
Material characteristic [ ] Isotropic [ ] Orthotropic

Elastic modulu 0.0
Poissons ratio 0.0
Specific heat 0.0
Mass density 0.0

Figure 5 The Edit Material Dialog



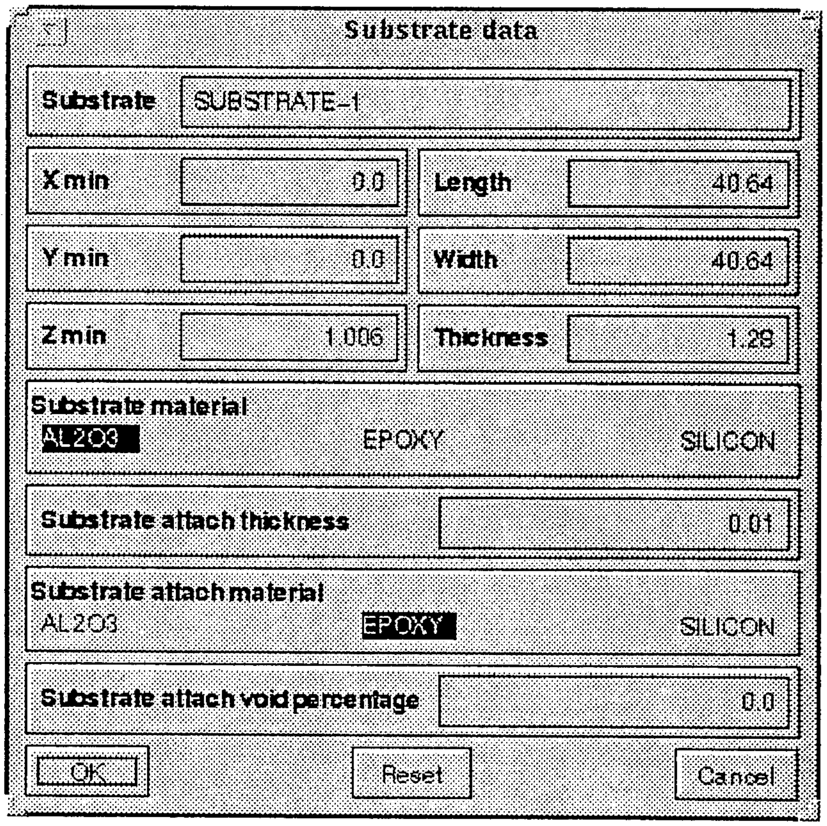
Package data

Package	PACKAGE-1		
X min	0.0	Length	40.64
Y min	0.0	Width	40.64
Z min	0.0	Thickness	1.006

Material: [✓ AL2O3] EPOXY SILICON

Base temperature: 30

Figure 6 The Edit Package Dialog



Substrate data

Substrate	SUBSTRATE-1		
X min	0.0	Length	40.64
Y min	0.0	Width	40.64
Z min	1.006	Thickness	1.29

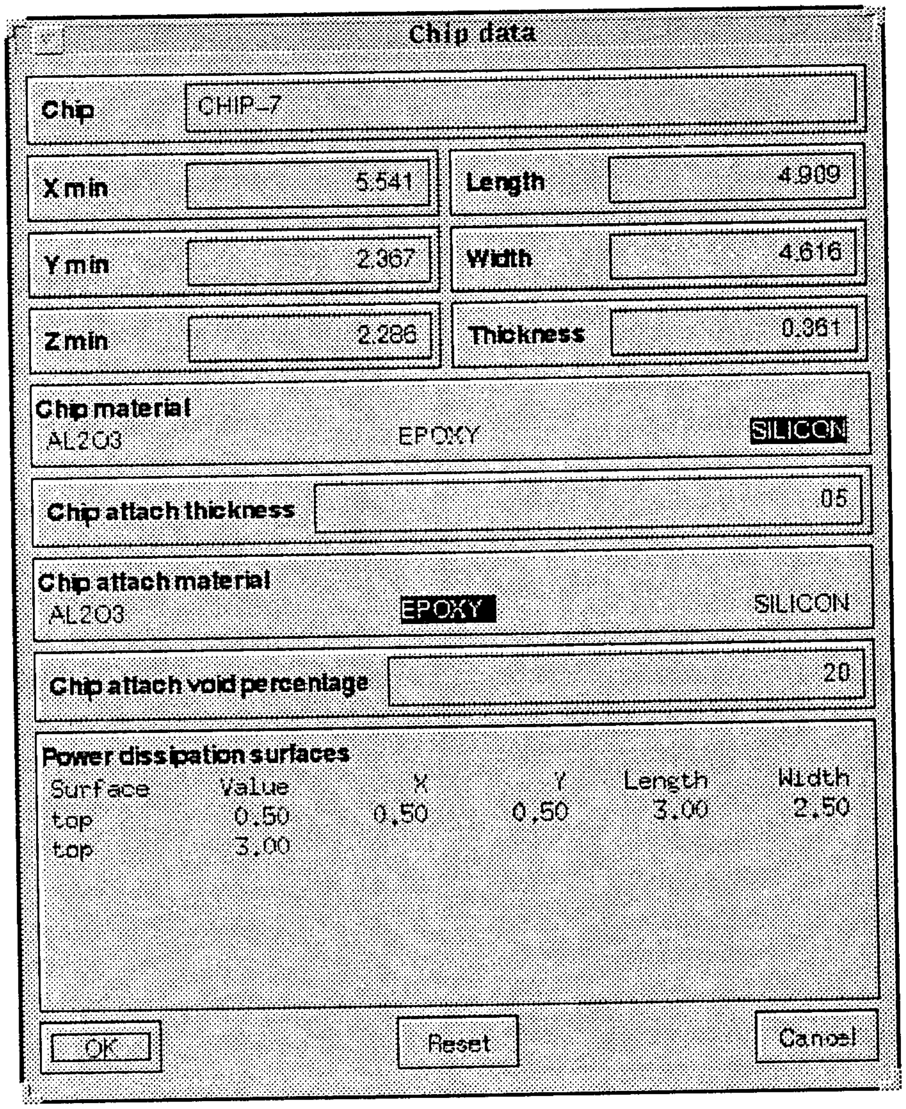
Substrate material: [✓ AL2O3] EPOXY SILICON

Substrate attach thickness: 0.01

Substrate attach material: AL2O3 [✓ EPOXY] SILICON

Substrate attach void percentage: 0.0

Figure 7 The Edit Substrate Dialog



Chip data

Chip	CHIP-7		
Xmin	5.541	Length	4.909
Ymin	2.367	Width	4.616
Zmin	2.286	Thickness	0.361

Chip material
AL2O3 EPOXY [✓ SILICON]

Chip attach thickness .05

Chip attach material
AL2O3 [✓ EPOXY] SILICON

Chip attach void percentage 20

Power dissipation surfaces

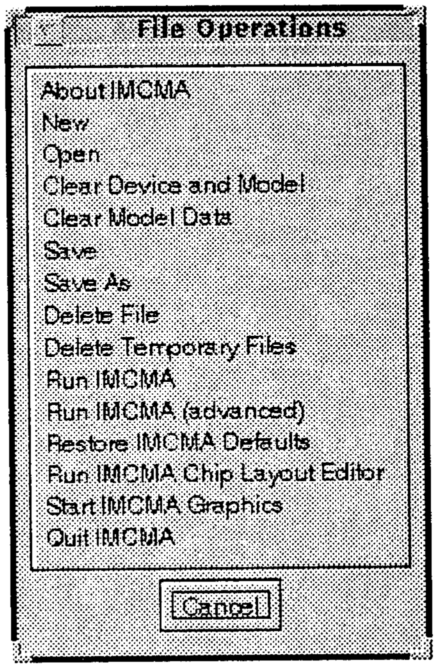
Surface	Value	X	Y	Length	Width
top	0.50	0.50	0.50	3.00	2.50
top	3.00				

Figure 8 The Edit Chip Dialog

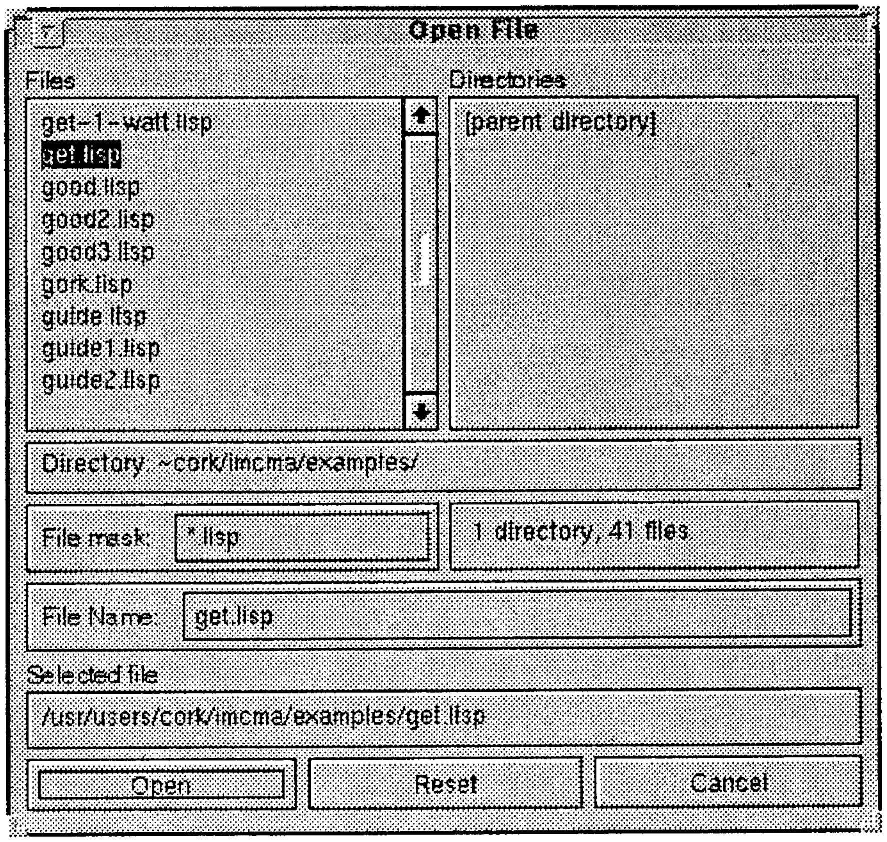
simply used as paired arguments to **reinitialize-instance**. GBB takes care of all the remaining bookkeeping associated with the material or component blackboard object.

Saving and restoring the device definition

Operations for loading or storing the device definition information are provided in the IMCMA file operations menu:



The **New** operation clears the blackboard of all defined materials and components in preparation for creating a new device definition. The **Open** operation allows the user to select an existing device-specification file for loading into a cleared blackboard. **Open** brings up a device-specification file dialog for selecting the file to be loaded:



get-1-watt.lisp
get.lisp
good.lisp
good2.lisp
good3.lisp
gork.lisp
guide.lisp
guide1.lisp
guide2.lisp

[parent directory]

Directory: ~cork/imcma/examples/

File mask: *.lisp

1 directory, 41 files

File Name: get.lisp

Selected file

/usr/users/cork/imcma/examples/get.lisp

Once an device has been defined or loaded, the user can perform an analysis (using the **Run IMCMA** operation) or edit and save the device definition using the **Save** or **Save As** operations.

Result Display Enhancements

Availability of the ChalkBox graphics toolkit allowed some significant enhancements to be made to the result-display facilities of the original IMCMA prototype. In the original prototype, three separate GBB graphics windows were used to display the top, front, and right side of the MCM device. The GBB graphics windows provide the ability to zoom in to specific regions, select individual or groups of blackboard objects (components, elements, nodes) for inspection, and so on.

One problem with the use of multiple windows in the original prototype is that the views are not synchronized and that the user can relocate (and even misplace) the individual views. Using the ChalkBox-based GBB graphics in GBB V3.0, we were able to create a single GBB-graphics window containing three *blackboard widgets*, each corresponding to one of the GBB graphics windows in the original prototype (Figure 9). Methods were written on these widgets so that

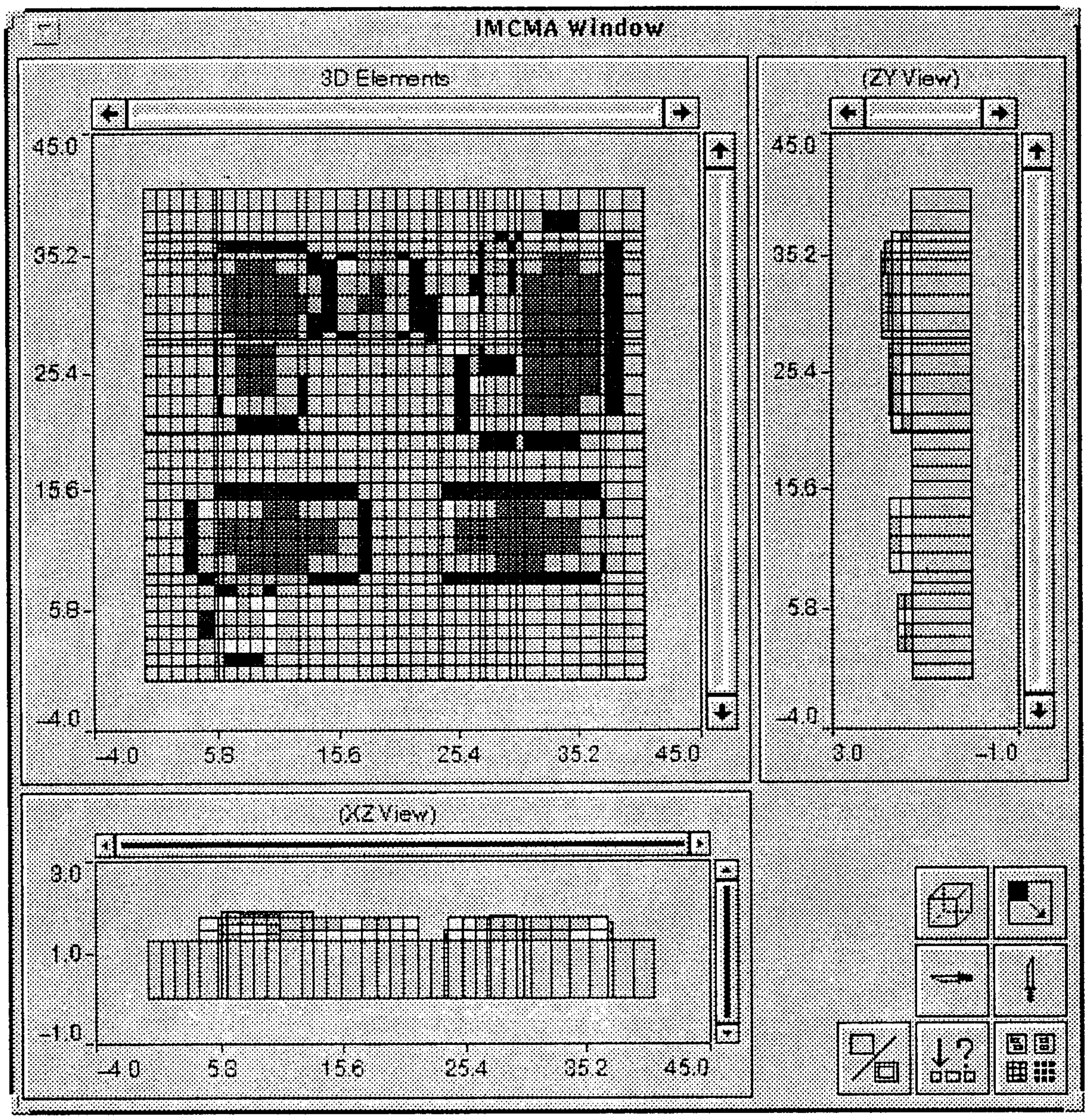


Figure 9 The IMCMA Window

operations performed on one (such as zooming in or scrolling) are propagated to the other widgets as well.

Along with the 3 blackboard widgets, command buttons for changing the display, filtering the display based on selected components, displaying interior "slices," showing component outlines, and resizing the display window were provided. The display selection button allows the display window to show the defined components, the modeled components, the 3d-mesh elements, or the 3d-mesh nodes (Figure 10).

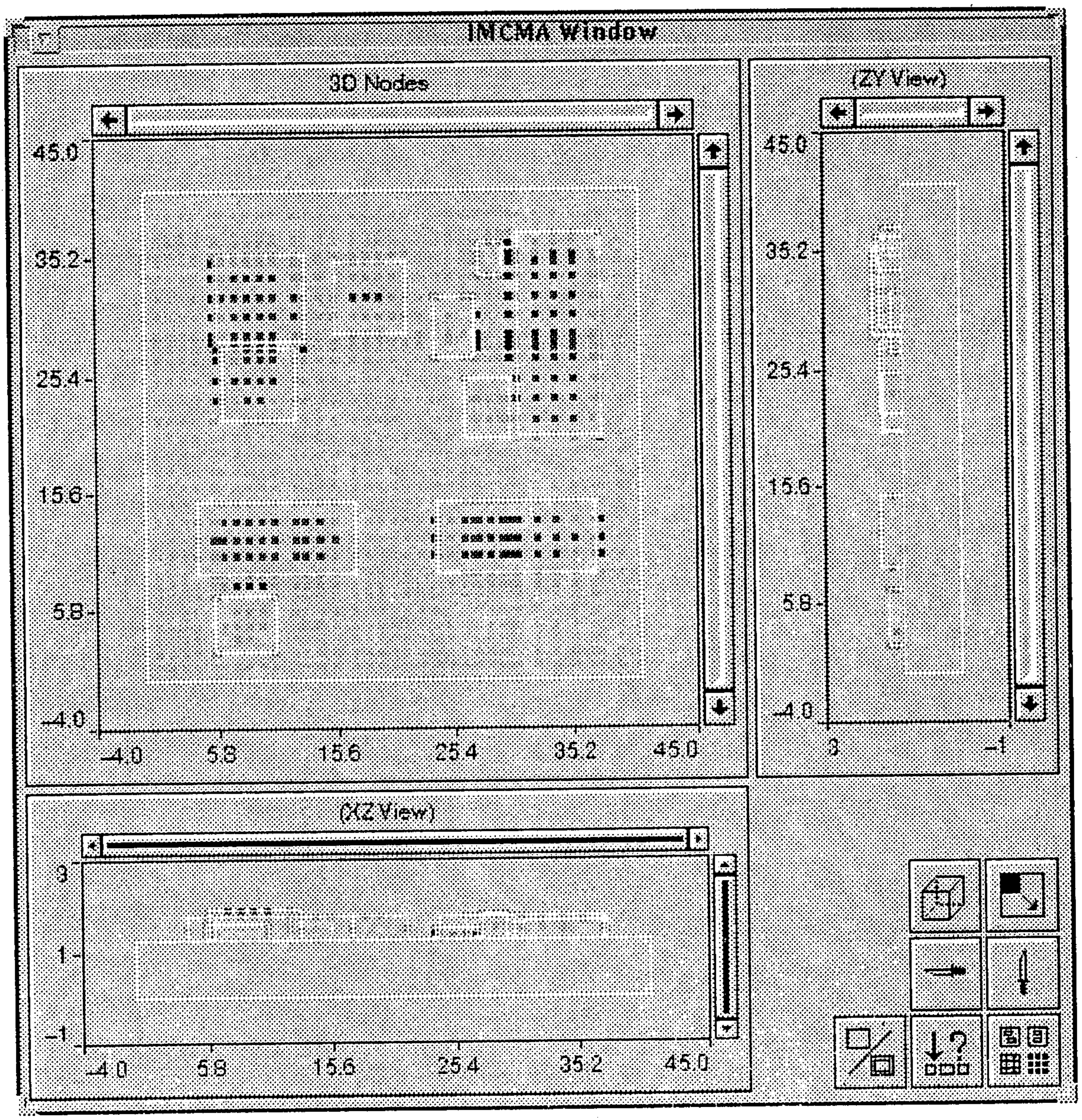


Figure 10 The IMCMA Window Displaying 3D Nodes

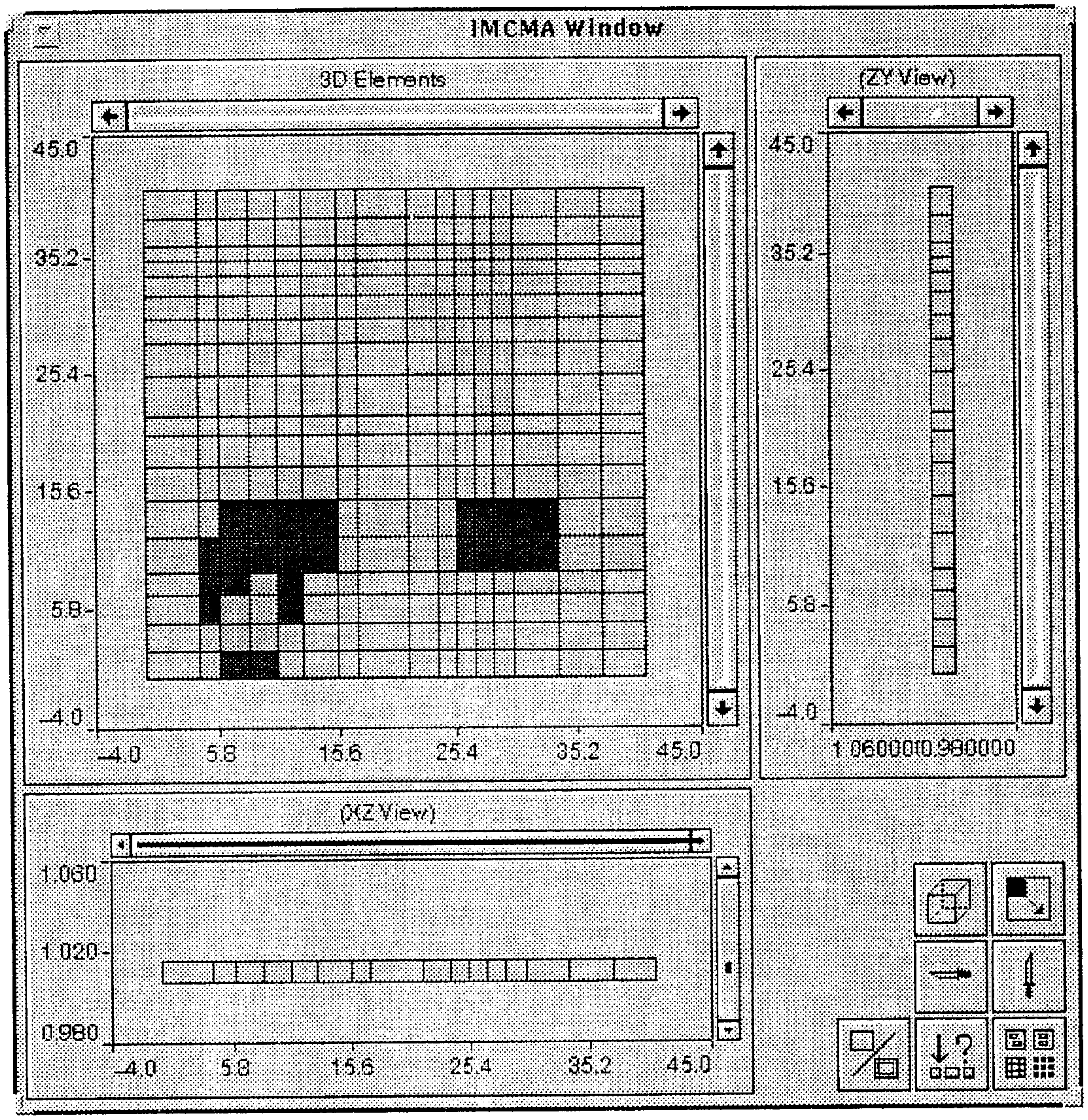


Figure 11 The IMCMA Window Displaying the Substrate-Attach Elements

Component filtering

The component filtering button provides a convenient, graphical means of showing the 3d-elements or 3d-nodes of only selected components. For example, this command button can be used to quickly view the thermal values of the 3d-elements in a substrate-attach layer (Figure 11).

Slicing

The "slice" buttons allow the user to interactively position a horizontal or vertical *slice line* on any of the three blackboard widgets. Only objects "sliced" by the slice line are then displayed on the other two

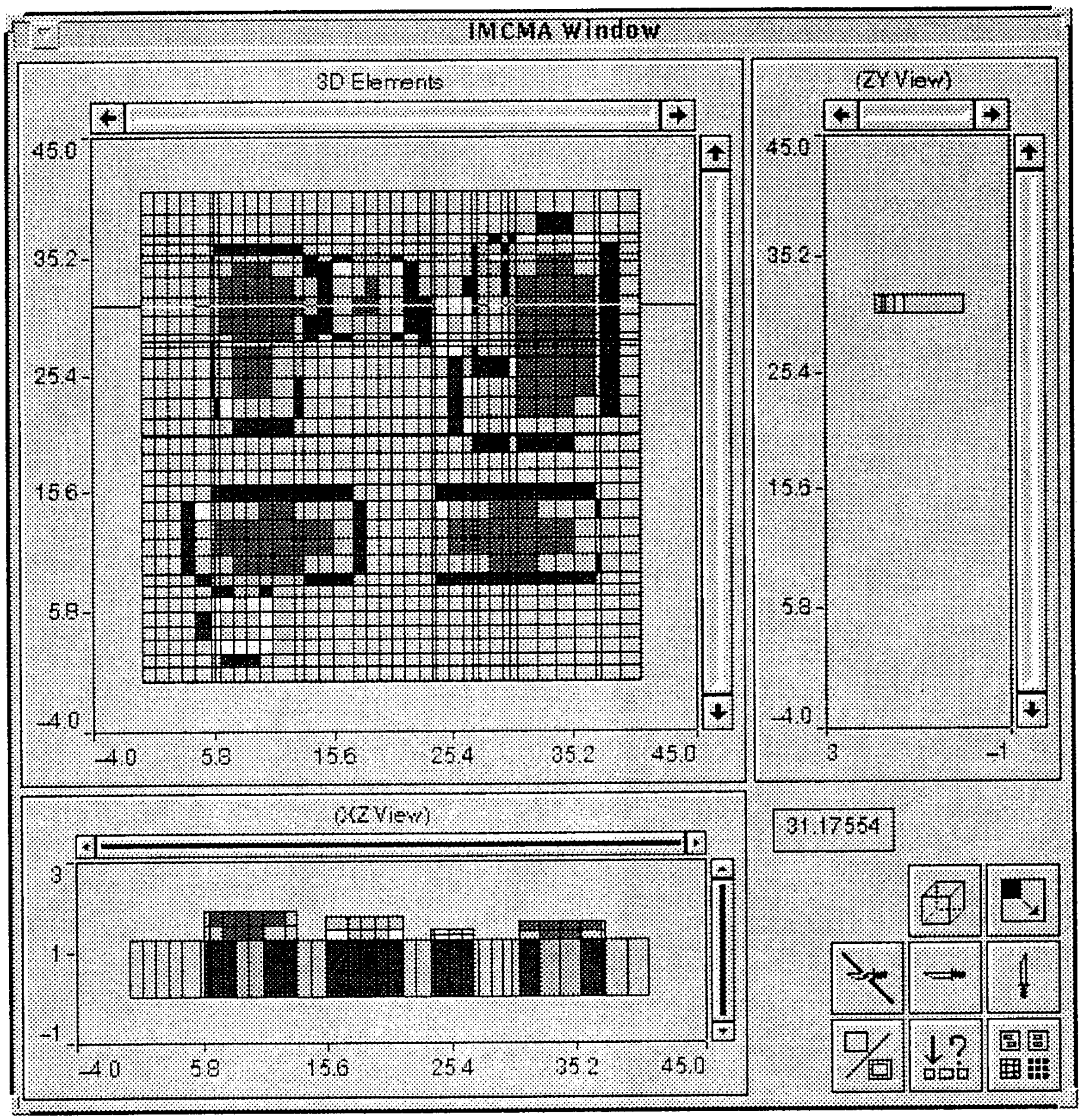


Figure 12 The IMCMA Window Displaying the "Slice" Operation

blackboard widgets (Figure 12). This facility is very useful for quickly viewing what is happening in the interior of an MCM device.

Temperature scale widget

A special ChalkBox temperature-scale widget was developed to display the temperature values associated with the thermal pseudo-colors used in the blackboard-widget displays. This widget automatically configures itself in horizontal (landscape) or vertical (portrait) mode according to its shape. It also displays temperature values in either Fahrenheit or

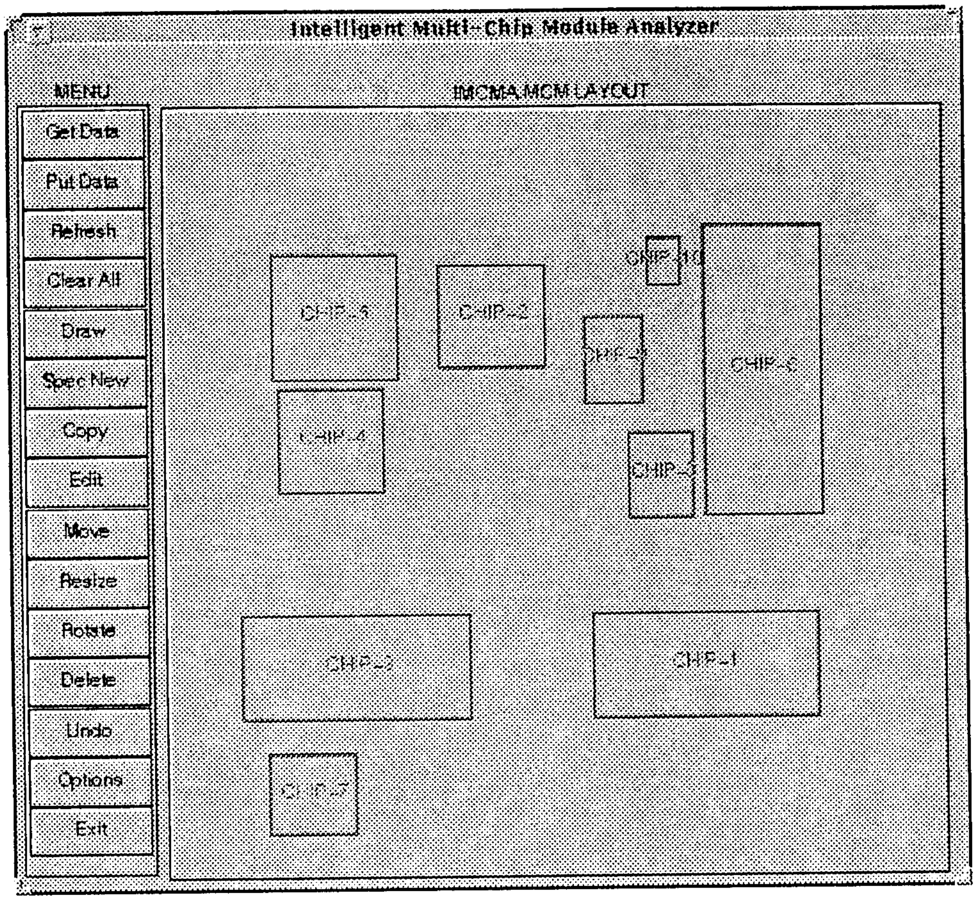


Figure 13 The Interactive IMCMA Chip Editor

Celsius. The temperature widget is easily customized and could be used to display stress values as well.

Interactive Chip-Layout Tool

In order to allow the user to conveniently enter and change the input data for processing, a "proof-of-concept" graphical chip-specification facility was developed. This facility allows the user to view the current XY positions of chips in the multichip module and to place and drag the chips as desired. Provision was also made to allow the user to directly enter the specifications from a data sheet. The editor is shown in Figure 13.

Representing the chips

Rather than directly manipulating the defined-device objects on the blackboard the interactive device layout tool uses a private blackboard (`mcm-editor`) and objects to represent the device and its chips. A blackboard-unit class called `chip-object` was defined to represent the individual chips. This class inherits from the defined-component unit class `chip` which was already defined in IMCMA.

By keeping the MCM editor blackboard separate from the `'(defined components)` space, it was possible to edit the chips and commit the changes *only* if the changes were satisfactory. GBB's advanced unit-retrieval capabilities made it easy to implement finding the chips that overlapped a given point in space. By mapping the mouse click coordinates onto the blackboard space, all chips that included that point could be found. Once found, the user could manipulate the chip in any desired way, either by dragging it with the mouse or directly specifying its new properties.

To support undo operations in the MCM Editor, a second unit class, `chip-temp`, was defined to represent the prior state of `chip-object` unit instances (see below). The organization of the blackboard database with the MCM Editor in operation is shown in Table 3.

Editing MCMs: Changing the database

Editing the layout and characteristics of a chip involves first locating the chip and then changing the database to reflect the desired changes. The desired changes may be a change in dimensions attributes such as position and size, or changes to other attributes which cannot be rendered graphically.

When the user indicates the desired chip by clicking the mouse over its graphical representation, the coordinates of the mouse click are translated to device coordinates (using the pixel-to-world mapping functions provided by ChalkBox). The chip is then located using GBB's blackboard retrieval facilities (Figure 14).

Chip relocation

A chip can be moved by clicking on its graphical represention with the mouse and dragging the chip to its new location. To achieve this, the chip is first located as described above. In this case, the offset of the mouse cursor within the chip is recorded, as this offset must be maintained as the mouse is dragged. The `move-rectangle` procedure creates the illusion of dragging the chip by erasing the previous position

```
Blackboard/Space Name       Contents
---------------------       --------
MCM-EDITOR
   CHIP-LAYOUT              9 Units: (9 CHIP-OBJECT)
   UNDO-CHIP-LAYOUT         10 Units: (10 CHIP-TEMP)
MODEL
   2D-MODEL
      COMPONENTS            Empty
      MAPMESH-2D-REGIONS    Empty
      MAPMESH-2D-LINES      Empty
      MAPMESH-2D-POINTS     Empty
      2D-ELEMENTS           Empty
      2D-NODES              Empty
   3D-MODEL
      COMPONENTS            Empty
      COMPONENT-3D-LINES    Empty
      COMPONENT-3D-POINTS   Empty
      COMPONENT-3D-SURFACES Empty
      3D-ELEMENTS           Empty
      3D-NODES              Empty
   MATERIALS                Empty
DEFINED
   COMPONENTS               14 Units: (1 DEVICE, 1 PACKAGE,
                                       1 SUBSTRATE, 1 ATTACH,
                                       10 CHIP)
   COMPONENT-2D-SURFACES    Empty
   COMPONENT-2D-LINES       Empty
   COMPONENT-2D-POINTS      Empty
   COMPONENT-3D-LINES       Empty
   COMPONENT-3D-POINTS      Empty
   COMPONENT-3D-SURFACES    Empty
   MATERIALS                3 Units: (3 MATERIAL)
LIBRARY
   LIBRARY-COMPONENTS       Empty
   LIBRARY-MATERIALS        Empty
```

Table 3 The blackboard database when editing an MCM

```
(defun FIND-A-CHIP (x y)
  (let ((chips-at-xy (find-units *chip-object-type*
                                 *chip-editor-path*
                                 '(:and
                                    (x :includes ,x)
                                    (y :includes ,y))))
        (bbtech-tools:sole-element chips-at-xy))))
```

Figure 14 The `find-a-chip` Function

(by drawing in XOR mode) and drawing a new representation at the new location (relative to the offset). While the mouse button remains depressed, the mouse location is sampled (using ChalkBox's `get-mouse-state` function) and if changed from the previous location, the chip is again erased using XOR and redrawn. Once the user releases the mouse button at the new location, the coordinates of the chip are updated in the `chip-object` unit instance.

Copying chips

Sometimes starting with a duplicate of an existing chip on the MCM is a convenient shortcut in defining a new chip. The MCM Editor provides a **Copy** operation that uses GBB's **copy-units** function to create a `chip-object` that has identical attributes to the chip being copied. Individual attributes of the copy can then be entered using the edit-chip dialog.

Undo

In any graphical editing environment, an undo capability is a must. The prototype chip-layout tool provides two levels of undo.

Because the editing session operates on a copy of the actual defined components, the user is free to ignore all changes made in an editing session by not saving the changes back to the defined-component blackboard. The user can also revert the session back to the original state (matching the defined-component blackboard) by reloading the chip data.

The last editing operation can also be undone. Each editing operation that changes the state of the `chip-layout` blackboard maintains maintains information on the `'(mcm-editor undo-chip-layout)` blackboard for the editor to the state prior to the operation.

4 Lessons Learned

As a result of this project, the prototype IMCMA system become substantially easier to use by an novice design engineer. In the process, we learned a number of things about the use of IMCMA, the Sandia finite-element meshing tools, and about application robustness and ease of use.

Modeling versus Reality

By explicitly separating the representation of the actual MCM device and materials from representations used in the modeling process, a number of potential problems were eliminated. It is now easy to remodel a device under different assumptions by merely changing the mapping from the defined component and material objects (the device specification) to the modeled components and materials. In the enhanced IMCMA prototype, designer engineers are now free to restart the modeling process after making changes to the modeled components and materials or to recreate the modeled components and materials from the original device definition objects—without reloading from a device-specification file. This capability provides the potential for easily implementing mechanized, iterative analysis approaches layered on the IMCMA prototype.

The separation also protects the carefully specified device and material objects from changes made during modeling. The user is free to save the current state of the defined device, knowing that only changes she has made to those objects will be reflected in the updated device-definition file.

Use of the Sandia Finite-Element Tools

Although the use of the FORTRAN-based Sandia codes in the IMCMA prototype system in the original IMCMA prototype architecture helped speed the initial development effort, their continued use is becoming more of a liability. The Sandia codes are not well-integrated into IMCMA due to the binary Exodus file data-exchange interface (rather than a complete API) and operator-command structure. The Sandia codes are inflexible and are not well suited to keeping track of meshing data through a number of tool invocations, requiring IMCMA to perform a great deal of bookkeeping and naming (numbering) conversions.

Given that much of the reasoning needed to generate the 3D mesh is being handled outside the Sandia tools and that a straightforward rectilinear mesh is being used, it may be time to consider replacing the Sandia codes. This possibility is discussed in the "Improvements for IMCMA" section, below.

Use of the ChalkBox Graphics Toolkit

By the end of this project, the ChalkBox graphics toolkit of GBB 3.0 was providing significant advantages in quickly constructing and updating the interactive graphic interfaces for data entry and result display. During the initial months of this project, however, we were using alpha and beta pre-releases of the ChalkBox product. In essence, we were among the first applications written in ChalkBox and, as such, discoverers of early weaknesses and bugs.

In retrospect, however, the choice of ChalkBox was a good one. Although we started with a complete design for the various menu-based interfaces for data input, these evolved throughout the effort as experience was gained using them in specifying real MCM devices. ChalkBox is based on a philosophy of dynamically laying out displays and menus at runtime, based on the items (widgets) being displayed and the real estate available for the various windows. Unlike a toolkit where widget positions are prespecified (even with a graphical editor), the use of ChalkBox allowed easy restructuring of displays without the need to manually reposition the various widgets.

Robustness and Ease of Use

As IMCMA became more graphical, users began to view it more as a "real" application rather than an experimental system. Errors and other problems that were previously resolved using the Lisp-listener-based command interface were no longer acceptable to the user. The use of the IMCMA system also grew substantially as it became easier to use. This additional use and willingness to experiment with IMCMA uncovered further weaknesses in the prototype architecture. Although not discussed explicitly in this report, a number of repairs and low-level enhancements were made to the IMCMA prototype in conjunction with (and as a result of) this effort.

The individual graphical dialogs and the overall dialog structure also evolved from a fairly complete initial design to a different organization and representation in response to user feedback on early versions. As a result, the graphical interface resulting from this project has been nicely tuned to the needs of users that have been using IMCMA to analyze real MCM devices. Such a "shakedown" period was important to the success of this effort, and the use of a flexible graphic environment (such as ChalkBox) allowed rapid responses to user feedback early in this project.

5 Improvements for IMCMA

Problems with the Sandia Tools

A number of problems stem from the use of the Sandia codes in the IMCMA system:

- **The Sandia codes are not well-integrated into IMCMA.** One of the requirements of the IMCMA system was that any off-the-shelf codes used in the system would be used without modification. For use as IMCMA KSs, this required that the codes use ASCII text files and piped commands to receive input information, and that IMCMA read FORTRAN-generated binary output files. Approximately one-half of the total processing time required for an initial analysis of an MCM device is expended in these interfaces.
- **The Sandia codes add minor capabilities.** Because a significant amount of reasoning about how to generate an appropriate 3D mesh is already performed within IMCMA, the Sandia codes perform little functionality beyond the bookkeeping required to generate a 3D mesh from detailed specifications. Developing GBB-based IMCMA KSs to directly generate the 3D mesh from these specifications would be relatively straightforward.
- **The Sandia codes are inflexible.** A significant amount of code was added to IMCMA to work within the limits of the Sandia codes. Because of limitations in the ways the Sandia codes can be used, representations of "pseudo-components," components, and materials must all be used at appropriate times during the mesh-generation process. At this point, it is estimated that the amount of such additional code is about as much as the amount of code that would be required to provide the Sandia-code functionality directly within IMCMA.
- **The Sandia codes add a system-administrative burden.** The Sandia codes are large and require an extensive library of routines to install, build, and maintain. Unlike the GBB-based IMCMA code, porting these codes to a new platform requires considerable effort.[3] Elimination of the Sandia codes would make IMCMA much more self contained.

[3]For example, the GBB-based portion of IMCMA was ported from a DECstation to a Sun Sparc workstation in a few hours. Installing the Sandia codes on the Sun Sparc required a number of weeks.

- **The Sandia codes are not universally available.** The Sandia codes are available for governmental and educational users only. They are not readily available to commercial users. This makes the IMCMA system unavailable for evaluation by many of the manufacturers and developers of multichip modules.

One of the advantages of using a blackboard architecture in the IMCMA prototype system is the ability to replace the Sandia-based KS codes with equivalent or enhanced KSs. The rectilinear nature of the finite-element mesh-generation approach used in IMCMA presents an opportunity for a implementing simpler and faster mesh-generation KSs. For example, now that the 3D mesh is stored directly on the IMCMA blackboard, it might be reasonable to consider developing GBB-based mesh-generation KSs that use the efficient blackboard-object-retrieval facilities of GBB during mesh generation. Such an effort would resolve all the problems noted above, resulting in an IMCMA system that is smaller, faster, more portable, more available, and more self contained.

3D Display of Analysis Results

One of the original goals of using the Sandia Tool set was the use of their analysis display routines (particularly BLOT) in IMCMA. Although it is now possible to display the analysis results using these tools, doing so has been cumbersome.

One difficulty was in adding the analysis results to the Exodus file. The difficulty of writing FORTRAN-compatible binary files from within Common Lisp required some tedious coding.

A bigger difficulty was using BLOT interactively from within IMCMA. BLOT is essentially an operator-driven display tool with a character-based command structure. Typically BLOT users enter commands, see the results, and enter modified commands until they are happy with the resulting plot.

Attempts to use BLOT from within IMCMA by determining appropriate command sequences proved unpredictable, due to unanticipated interactions among the various BLOT commands. Thus, at the end of this effort, BLOT use remains primarily an off-line, post-analysis utility.

Because, it is important so to provide the design analyst with easily understood analysis results, adding other ways of presenting selected information for expert and novice analysts would greatly improve the ease of use of the IMCMA prototype. Although the choice of GBB as the implementation environment for IMCMA provides built-in capabilities to graphically present two-dimensional views of multidimensional information, most design analysis are accustomed to three-dimensional contour plots of analysis results. We recommend that further emphasis on this aspect of result display be considered.

6 Summary of Accomplishments

In this effort, we extended the architecture to represent finite-element mesh objects and results directly on the blackboard, provided separate device description and device "as modeled" representations, developed graphical facilities for creating, modifying, copying, and deleting MCM-device components and material specifications, and redesigned and implemented an improved interactive result-display facility. As a result of this effort, the IMCMA prototype system can now be more easily used by casual users.

7 References

[1] W. J. Bocchi, J. A. Collins, and D. J. Holzhauer. Thermal stress analysis of integrated circuits using finite element methods. Technical Report RADC-TR-84-100, Rome Air Development Center, Rome, NY, April 1984.*

[2] W. J. Bocchi. Finite element modeling and thermal simulations of transistor integrated circuits. Technical Report RADC-TR-89-176, Rome Air Development Center, Rome, NY, October 1989.

[3] J. H. Lau, D. W. Rice, and P. A. Avery. Elastoplastic analysis of surface mount solder joints. *IEEE Transactions on Components, Hybrids and Manufacturing Technology*, CHMT-10(3):346–357, September 1986.

[4] P. A. Engel and C. K. Lim. Stress analysis in electronic packaging. *Finite Element Analysis and Design*, 4(1):9–18, June 1988.

[5] J. H. Lau and C.G. Harkins. Thermal stress analysis of SOIC packages and interconnections. *IEEE Transactions on Components, Hybrids and Manufacturing Technology*, 11(4):380–389, 1988.

[6] J. C. Glaser and M. P. Juaire. Thermal and structural analysis of a PLCC device for surface mount processes. *Journal of Electronic Packaging*, 3(3):172–178, September 1989.

[7] S. Kawai. Structural design of plastic ic packages. *JSME International Journal, Series 1—Solid Mechanics, Strength of Materials*, 32(3):320–330, July 1989.

*Although this report references the limited document noted above, no limited information has been extracted. Distribution authorized to USGO Agencies and their contractors; critical technology - to protect information which advances the state-of-the-art or new technology of significant military application.

[8] J. H. Lau. Thermal stress analysis of SMT PQFP packages and interconnections. *Journal of Electronic Packaging*, 3(1):2–8, March 1989.

[9] George Turklyyah and Stevern J. Fenves. Feasibility study of a knowledge-based finite-element modeling assistant. Final report, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, February 1988.

[10] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[11] Daniel D. Corkill. Blackboard systems. *AI Expert*, 6(9):40–47, September 1991.

[12] Daniel D. Corkill. An architecture for intelligent multichip module reliability analysis. Final Report RL-TR-94-71, Rome Laboratory, Griffiss Air Force Base, New York, April 1984.

Appendices

A IMCMA Test MCM Device-Description File

The device-description file for the Test MCM example (with the chips located on top of the substrate) is shown below. This is the same TEST MCM device used in the IMCMA 1.0 report [12].

```
;;;; -*- Mode:COMMON-LISP; Package:IMCMA; Base:10 -*-
;;;; *-* File: DIAMOND: /usr/users/cork/newimcma/examples/get.lisp *-*
;;;; *-* Edited-By: Cork *-*
;;;; *-* Last-Edit: Wednesday, August 31, 1994  17:14:10 *-*
;;;; *-* Machine: GRANITE (Explorer II, Microcode 489) *-*

;;;; ***************************************************************************
;;;; ***************************************************************************
;;;; *
;;;; *                            TEST MCM DEVICE
;;;; *
;;;; ***************************************************************************
;;;; ***************************************************************************
;;;
;;; Written by:  Dan Corkill
;;;              Blackboard Technology Group, Inc.
;;; Modified by: Prasanna Katragadda
;;;              ME, UMASS,  04/01/93
;;; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;;;
;;;  11-18-92 File created.
;;;
;;; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

(in-package "IMCMA")

;; This must come first:

(define-device "TEST MCM"
  :filename "test-mcm"
  :size (40.64 40.64 2.955))
```

```
;; Materials come next:

(defmaterial :silicon
  :min-error-tolerance .1
  :max-error-tolerance .1
  :tk (0.1256 0.1256 0.1256)
  :alpha (0.233e-05 0.233e-05 0.233e-05)
  :beta 0
  :reference-temperature 0)

(defmaterial :Al2O3
  :min-error-tolerance .1
  :max-error-tolerance .1
  :tk (0.025 0.025 0.025)
  :alpha (0.8e-05 0.8e-05 0.8e-05)
  :beta 0
  :reference-temperature 0)

;; Now the device:

;; The aluminum-oxide carrier:

(defcomponent :SUBSTRATE-1 :substrate
  :size (40.64 40.64 1.27)
  :prescribed-temperature-surfaces ((:bottom 30))
  :material :Al2O3)

;; The Chips:

(defcomponent :CHIP-1 :chip
  :size (13.0010 5.9890 .516)
  :x (+ (/ 40.64 2.0) 10.1451)
  :y (+ (/ 40.62 2.0) -8.4118)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-2 :chip
  :size (6.0000 6.0000 .516)
  :x (+ (/ 40.64 2.0) -1.8409)
  :y (+ (/ 40.62 2.0) 11.6080)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-3 :chip
  :size (12.9887 5.9890 .516)
  :x (+ (/ 40.64 2.0) -9.6919)
  :y (+ (/ 40.62 2.0) -8.4363)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)
```

```
(defcomponent :CHIP-4 :chip
  :size (6.0000 6.0000 .516)
  :x (+ (/ 40.64 2.0) -11.0530)
  :y (+ (/ 40.62 2.0) 4.4870)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-5 :chip
  :size (7.2430 7.2440 .669)
  :x (+ (/ 40.64 2.0) -10.8227)
  :y (+ (/ 40.62 2.0) 11.6080)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-6 :chip
  :size (6.7090 16.7800 .429)
  :x (+ (/ 40.64 2.0) 13.6987)
  :y (+ (/ 40.62 2.0) 8.4359)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-7 :chip
  :size (4.9090 4.6160 .361)
  :x (+ (/ 40.64 2.0) -12.3240)
  :y (+ (/ 40.62 2.0) -15.6350)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-8 :chip
  :size (3.8500 4.9500 .480)
  :x (+ (/ 40.64 2.0) 7.8190)
  :y (+ (/ 40.62 2.0) 2.4290)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

(defcomponent :CHIP-9 :chip
  :size (3.4190 5.1230 .264)
  :x (+ (/ 40.64 2.0) 5.0069)
  :y (+ (/ 40.62 2.0) 9.0850)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)
```

```
(defcomponent :CHIP-10 :chip
  :size (2.0090 2.8490 .567)
  :x (+ (/ 40.64 2.0) 8.0830)
  :y (+ (/ 40.62 2.0) 14.7080)
  :z (+ 1.27)
  :xy-alignment :centered
  :prescribed-flux-surfaces ((:top 0.08))
  :material :silicon)

;;; ----------------------------------------------------------------------
;;;   End of File
;;; ----------------------------------------------------------------------
```

B Running IMCMA

This appendix provides a quick overview of the steps used to run the IMCMA prototype system.[4] The IMCMA graphical user interface (GUI) is activated by running the function `(imcma-gui)`.[5] When the IMCMA GUI is running, the IMCMA *main menu* will be visible:

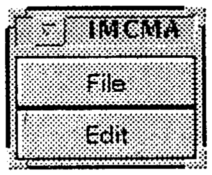


Figure 15 The IMCMA Main Menu

Selecting the **File** button brings up the following *file menu*:

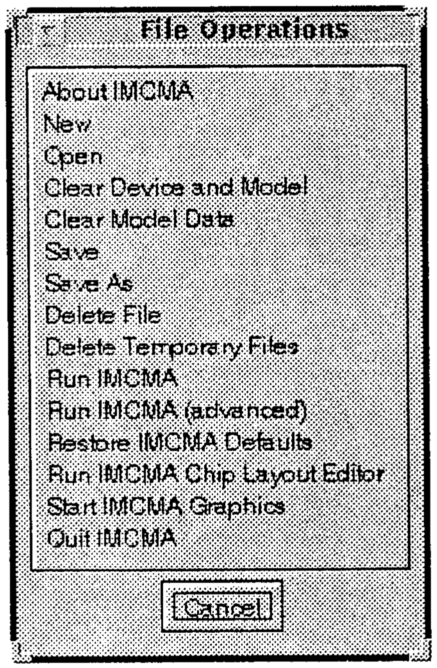


Figure 16 The File Operations Dialog

[4]This appendix assumes the IMCMA system has been correctly installed and loaded into GBB.

[5]Depending upon how IMCMA has been installed, this function may be automatically called as part of starting the IMCMA/GBB image.

Selecting the **Open** item brings up the *Open File* dialog (Figure 17) for selecting an existing device definition file. Once the desired file has been selected and the **Open** button clicked with the mouse, the device definition will be loaded into IMCMA.

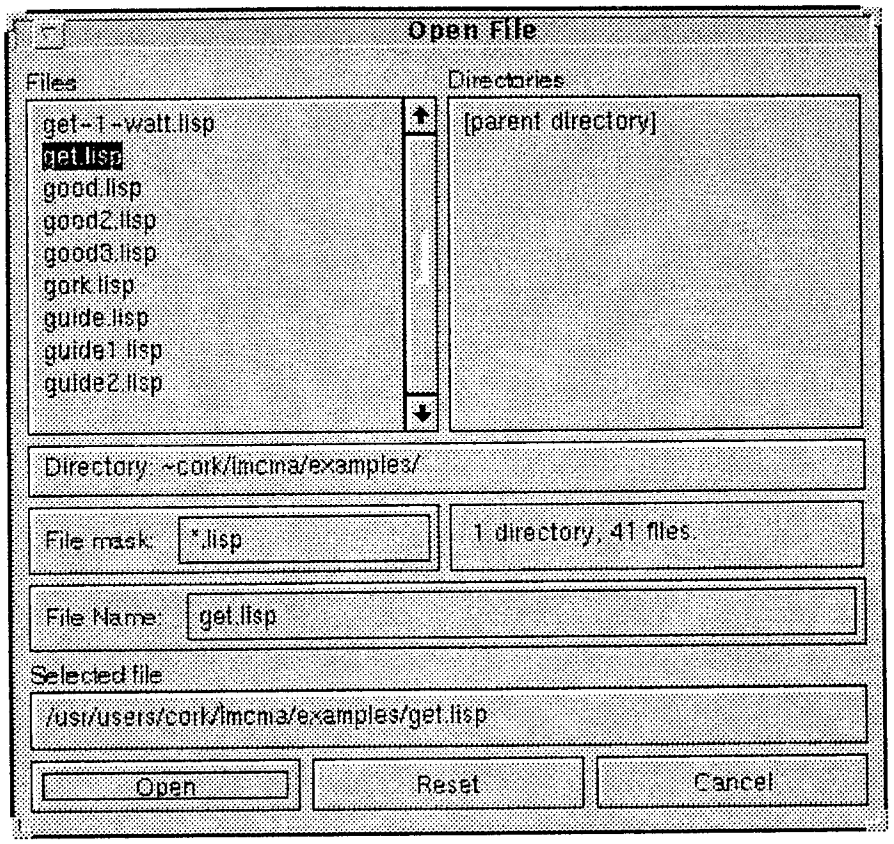


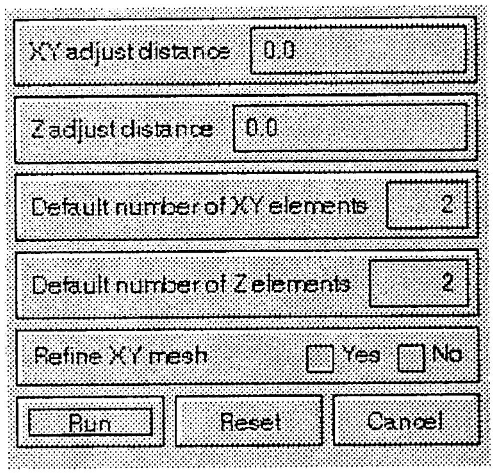
Figure 17 The Open File Dialog

At any point, the IMCMA graphics facility can be started using the **Start IMCMA Graphics** item in the **File** menu. This will start in the standard 2D display mode.[6]

To perform an IMCMA analysis, use the **Run IMCMA** item in the **File** menu. This will bring up the following dialog:

Clicking the Run button initiates IMCMA analysis of the device.

[6]The **IMCMA Mouse Documentation** window provides help in determining what can be done using the IMCMA graphics facility.



XY adjust distance 0.0

Z adjust distance 0.0

Default number of XY elements 2

Default number of Z elements 2

Refine XY mesh Yes No

Run Reset Cancel

Figure 18 The Run IMCMA Dialog

C Glossary

Blackboard (GBB)	A space or another GBB blackboard
blackboard widget (GBB)	A ChalkBox display for interacting with a two-dimensional view of blackboard data
ChalkBox	Product trademark for Blackboard Technology Group's graphic interface toolkit
device-definition file	text-based file containing component and material specifications for an MCM device
DOVE	UMass's Design of Virtual Experiments shell written in conjunction with IMCMA
GBB	Product trademark for Blackboard Technology Group's generic blackboard framework
FEECAP	UMass finite-element analysis package
IMCMA	Intelligent Multichip Module Analyst system
KS	Knowledge source
KSA	An activation of a knowledge source
Link	A bidirectional relationship between two GBB blackboard units
MCM	Multichip module
Space (GBB)	A blackboard level or plane
UMass	University of Massachusetts
Unit (GBB)	A blackboard object

*U.S. GOVERNMENT PRINTING OFFICE: 1996-710-126-20085

MISSION OF ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.